OPTIMAL LOCATIONS ON NETWORKS— A PSEUDO-BOOLEAN APPROACH

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

BY
UDAI KRISHNA GARG

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY KANPUR

SEPTEMBER 1972

TO MY MOTHER

CERTIFICATE

Certified that this work on " OPTIMAL LOCATIONS ON NETWORKS - A PSEUDO-BOOLEAN APPROACH " by U.K. Garg has been carried out under my supervision and that this has not been submitted elsewhere for a degree.

Narsingh Deo

Dr. Narsingh Deo
Associate Professor
Computer Center
Department of Electrical Engineering
Indian Institute of Technology Kanpur

ABSTRACT

In this dissertation, algorithms ensuring optimal results for following three problems concerning locations on networks are given.

(1) Finding all m-vertex median of a graph.

(2) Finding all modified m-vertex median of a graph.

(3) Finding the minimum number of emergency service facilities required for a network of destinations under certain assumptions.

For the first two problems only heuristic methods or complete enumeration are available. To find the optimal results for the above problems, the distinct approach adopted in this dissertation is to reduce the above problems to minimization of nonlinear pseudo-Boolean objective functions without any constraints.

Basic algorithm is available to minimize the unconstrained, nonlinear pseudo-Boolean functions.

Computer codes are developed for the algorithms proposed here and computational attributes of various algorithms are compared.

ACKNOWLEDGEMENTS

I am grateful to Dr. Narsingh Deo for his able guidance throughout the course of this work.

I wish to put on record my deep sense of indebtedness to Dr. S.K. Gupta, Dept. of Maths., for his valuable suggestions, appreciation and encouragement in the moments of depression.

I thank Mr. K. Ch. Mohan Rao for correcting mistakes and Mr. S.Kumar for excellent typing.

Udai Garg

CONTENTS

		Page
CERTIFICATE		
ABSTRACT		
ACKNOWLEDGEMENTS		
CHAPTER I	INTRODUCTION	1
CHAPTER II	A SURVEY OF EXISTING ALGORITHMS	4
	2.1 The m-Vertex Median	4
	2.1.1 Maranzana's method	7
	2.1.2 One optimal or vertex substitution method	8
	2.2 The Modified m-Vertex Median	13
	2.2.1 Modified Maranzana's method	14
	2.2.2 Modified One optimal or vertex substitution method	16
	2.3 Finding Minimum Number of Emergency Service Facilities	16
CHAPTER III	BASIC CONCEPTS OF PSEUDO-BOOLEAN PROGRAMMING	18
	3.1 Solutions of Linear Pseudo-Boolean Equalities	19
	3.2 Solutions of Linear Pseudo-Boolean Inequalities	23

			Page
	3.3	Determination of Characteristic Function in Various Cases	29
	3.4	Pseudo-Boolean Form of Characteristic Function	30
	3.5	Minimization of Unconstrained, Nonlinear Pseudo-Boolean Functions	31
CHAPTER IV	PSEUDO-BOOLEAN FORMULATION		35
	4.1	The m-Vertex Median	35
	4.2	Finding Minimum Number of Emergency Service Facilities	43
	4.3	The Modified m-Vertex Median	47
CHAPTER V	COMPUTATIONAL ATTRIBUTES		51
	5.1	Coding of Basic Algorithm	51
	5.2	Comparison of Computational Attributes	63
REFERENCES			67

CHAPTER I

INTRODUCTION

Determining good locations for sources on a network has received a reasonable amount of attention in the last decade. Some of the important problems, regarding optimal locations of these sources are:

(1) The problem of supplying n destinations from m sources has been attacked with a variety of assumptions and methods. If both sources (with given capacities) and destinations (with given demands) are at fixed locations, then this becomes a standard transportation problem.

But here the task is to determine the locations of the sources so as to minimize the transportation cost, when the locations for the destinations are given with certain assumptions. If the destinations consist of fixed vertices on a network, but the sources may lie any where on the network-edges and the destination demands are fixed, while the source capacities are unconstrained, it has been shown by Hakimi[2] that the problem resolves itself (by taking transportation costs proportional to distance) into finding the generalized absolute median of the corresponding graph to the network in which both vertices and the edges have weights. The weight of a vertex represents a destination demand and the weight

of an edge represents the shortest distance between the corresponding vertices in the network. Hakimi also demonstrated that there will exist such an m - median that includes only vertices of the graph. Thus a solution to this problem will correspond to the case where both destinations and the sources lie on the vertices of a network, a situation like that investigated for fixed destination demands and the unconstrained source capacities by Maranzana[8].

(2) While locating the sources, an important consideration is the maximum distance that seperates a destination from its nearest source. In first problem, by imposing a limit on the maximum distance that seperates a destination from its nearest source, we will have to solve the first problem under n new constraints (one for each destination and hence in corresponding graph G one for each vertex) to meet the specified distance standards. This is exactly what the second problem is and has been named as 'Modified m-vertex median of a graph'[10].

(3) In the third problem we have to find the minimum number of sources required* to meet the distance standards or constraints (i.e. limit on the maximum distance that seperates a destination from its nearest source) of each of the destinations[10]. As such this problem is especially applicable to the location of emergency service facilities (consider sources as service facilities here) such as fire stations,

*The assumptions made are same as in the previous problems.

although one may equally well apply it to the location of ordinary services, such as schools, libraries etc.

The location of fire stations might be approached according to the structure just described. The limit on response time/distance (as time may be taken proportional to distance) is imposed to ensure that no more than a specified time period s_j will elapse before a response will occur to any call at jth destination. The desired solution to this problem locates the minimum number of fire stations that satisfies the response time requirements.

CHAPTER II

A SURVEY OF EXISTING ALGORITHMS

In this chapter, a survey of the existing heuristic methods is done for the first problem. The heuristic methods for the first problem are modified so that they can be used for the second problem. Also the optimal algorithms available for the third problem are mentioned.

2.1 The m-Vertex Median[2]

Consider a graph G with weighted vertices and edges. The notations used are as follows:

- V – Vertex set of G having n vertices $v_1, \ldots, v_n$.
- x_j – Weight of the vertex v_j
- d_{ij} – Weight of the edge (i, j) i.e. the shortest distance between vertex v_i and the vertex v_j.
- D – Distance matrix (symmetric) with d_{ij}'s as the elements.

If in the graph G, all vertices have identical weights, a vertex median v_k will be that vertex for which the sum of the elements in the corresponding column of D is minimized. That is, let

$$d_j = \sum_{i=1}^{n} d_{ij} \text{ (for } j = 1, \ldots n) \tag{2.1}$$

then v_k is the vertex median if and only if

$$d_k = \min (d_1, \ldots, d_n) \tag{2.2}$$

If in the graph G, the vertices have unequal weights, then it is necessary to redefine the distance matrix. Let X be the nth order diagonal matrix with the vertex weights on the diagonal. Now the weighted distance matrix of G is defined by

$$H = DX = \left[h_{ij} \right] = \left[d_{ij}\, x_j \right] \tag{2.3}$$

Matrix H is no longer symmetric. Each h_{ij} represents the weighted distance associated with vertex v_j if v_i were the unique source. The vertex median is defined as above in Eqn.* (2.1) and Eqn. (2.2) but over the matrix H.

A generalization of vertex median follows logically from Eqn. (2.1) and Eqn. (2.2). Let V_m be a subset of V containing exactly m vertices of G. For an n vertex graph there will be $\binom{n}{m}$ possible subsets of cardinality m and we arbitrarily index them by V_m^p $(p = 1,2, \ldots \binom{n}{m})$. For each subset we may construct a submatrix H_m^p of H by adjoining all rows of H for which the corresponding vertices are contained in V_m^p. H_m^p is of order mxn and describes precisely the sources and the associated weighted distances for every destination if the set of sources is limited to vertices in

---

* Henceforth whenever we use Eqn., we mean either equality, inequality, or system of inequalities and/or equalities.

V_m^p . Since it is assumed that sources have no capacity constraints, each destination v_j will be served by that source v_k in V_m^p for which h_{kj} is a minimum, i.e.

$$h_{kj} \leq h_{ij} \; , \quad v_i \in V_m^p \tag{2.4}$$

The total weighted distance h_p for the set V_m^p of sources will be the sum of column minima of H_m^p.

$$h_p = \sum_{j=1}^{n} h_{kj} \tag{2.5}$$

where k refers to the source for which h_{ij} is minimized. An m vertex median of G is now defined as same V_m^p such that

$$h_p^* = \min \left(h_1, h_2, \ldots h_{\binom{n}{m}}\right)$$

$$\text{so that } h_p^* \leq h_p \quad \left(p = 1, 2, \ldots \binom{n}{m}\right) \tag{2.6}$$

The m-vertex median is not necessarily unique. Clearly it defines a set of sources which is in some sense closest to all destinations.

So far a completely satisfactory method for finding the m-vertex median of a graph is not available. However, two heuristic approaches available till now are described below.

2.1.1 Maranzana's method[8]

The algorithm starts with an arbitrary selection of a subset V_m containing m vertices and partitions the graph into m subsets such that to each vertex in V_m, one subset of these m subsets is associated. This is accomplished by associating each vertex in G with its nearest vertex in V_m. Next the center of gravity* of each set in the partition is determined and the original vertices in V_m are replaced by these vertices. The process is repeated until the vertices in V_m do not change. A formal statement of algorithm follows:

Algorithm

Step 1: Arbitrarily select m vertices in V and assign these vertices to the variable array, v_{y_i} appearing in Step 2.

Step 2: Associated with this array of m vertices, v_{y_1}, v_{y_2}, ..., v_{y_m} determine a corresponding

---

*Center of gravity: Vertex v_{k*} will be a center of gravity of a subset V' of V if

$$\sum_{v_j \in V'} h_{k*j} \leq \sum_{v_j \in V'} h_{kj} \qquad \text{for all } v_k \in V$$

partition of V, P_{y_1}, P_{y_2}, ..., P_{y_m} by putting

$$P_{y_i} = \{ v_k \mid h_{y_i k} \leq h_{y_j k} \} \text{for all } y_j$$

If a vertex is equidistant from more than one v_{y_i}, a decision is required relative to placement of the vertex. To break the tie, put the vertex in set associated with v_{y_i} having smallest y_i among them.

Step 3: Determine the centre of gravity, C_{y_i} for each P_{y_i}.

Step 4: If $C_{y_i} = v_{y_i}$ for all i, computation is stopped and the current values of v_{y_i} and P_{y_i} constitute the desired solution; otherwise set $v_{y_i} = C_{y_i}$ and return to Step 2.

In Step 3 if the center of gravity is not unique, then choose one which has smallest subscript among them.

The algorithm has been shown to be monotonically[8] non-increasing with respect to the successive selection of P_{y_i} according to Step 2 and successive selection of value of C_{y_i} for P_{y_i} according to Step 3.

2.1.2 One Optimal or Vertex Substitution Method[9]

Consider the definition of the m vertex-median by Eqn. (2.5) and Eqn. (2.6). For each possible subset of

vertices V_m^p we may construct a submatrix H_m^p of H by adjoining m rows corresponding to the vertices v_j in V_m^p . The vertex in V_m^p with which any vertex v_j is associated is defined as that v_k such that

$$h_{kj} \leq h_{ij} \quad , \quad v_i \in V_m^p$$

This is the jth column minimum in H_m^p. The total weighted distance h_p for the pth subset V_m^p will be the sum of these column minima.

Suppose that we decide to replace one vertex v_i, in the subset V_m^p by another v_b. Several kinds of effect on the total weighted distance may occur.

If h_{ij} were not the jth column minima, then its replacement by h_{kj} might have several outcomes depending on whether

$$h_{tj} \leq h_{bj} \tag{2.8a}$$

or

$$h_{tj} \geq h_{bj} \tag{2.8b}$$

where h_{tj} is jth column minima of H_m^p.

In case of Eqn. (2.8a), the jth column contribution to h will be zero i.e. $j\Delta_{bi} = 0$.

In case of Eqn. (2.8b)

$$j \triangle bi = h_{bj} - h_{tj}$$

and

$$j \triangle bi \leq 0$$

If h_{ij} were the jth column minima, following will be observed:

$$h_{bj} \leq h_{ij} \tag{2.9a}$$

or

$$h_{ij} \leq h_{bj} \leq h_{sj} \tag{2.9b}$$

or

$$h_{ij} \leq h_{sj} \leq h_{bj} \tag{2.9c}$$

where h_{sj} is the weighted distance from vertex v_j to that vertex v_s for which

$$h_{ij} \leq h_{sj} \leq h_{s*j} \quad \text{for } v_{s*} \in V_m^p \quad \text{and } v_{s*} \neq v_i, v_s \tag{2.10}$$

In otherwords, h_{sj} is the second smallest jth column element in H_m^p. In case of Eqn. (2.9a), the jth column contribution to h from the substitution of v_b for v_i is now incremented by

$$j \triangle bi = h_{bj} - h_{ij} \tag{2.11}$$

and

$$j \triangle bi \leq 0$$

In case of Eqn. (2.9b), the jth column contribution to h is incremented by

$$
\begin{aligned}
{}_j\Delta_{bi} &= h_{bj} - h_{ij} \\
\text{and} \qquad & \\
{}_j\Delta_{bi} &\geq 0
\end{aligned}
\tag{2.12}
$$

In case of Eqn. (2.9c), the jth column contribution to h is incremented by

$$
\begin{aligned}
{}_j\Delta_{bi} &= h_{sj} - h_{ij} \\
\text{and} \qquad & \\
{}_j\Delta_{bi} &\geq 0
\end{aligned}
\tag{2.13}
$$

whether it is worth substituting vertex v_b for v_i depends upon the net effect of the increments summed over all columns

$$\Delta_{bi} = \sum_{j=1}^{n} {}_j\Delta_{bi} \tag{2.14}$$

Substituting v_b for v_i reduces total weighted distance only if

$$\Delta_{bi} < 0 \tag{2.15}$$

These observations direct us to develop a One optimal algorithm as follows.

<u>Algorithm</u>

Step 1: Select some initial vertex subset V_m. For convenience in expression let it contain vertices v_1, v_2, ..., v_j, ..., v_m.

Step 2: Compute the total weighted distance h_m for the system.

Step 3: Select some vertex, v_b, not in the subset V_m.

Step 4: For each vertex v_i in V_m substitute v_b and compute Δ_{bi}.

Step 5: Find that vertex v_k in V_m, such that

$$\Delta_{bk} < 0 \qquad (2.16)$$

and $$\Delta_{bk} = \min \Delta_{bi} \quad \text{for } (i = 1, 2, \ldots n)$$

Step 6: If a vertex satisfying the Eqn. (2.16) can be found, substitute v_b for v_k in the subset V_m, label the new subset so formed as V_m' and compute h_m'. If no vertex satisfies the Eqn. (2.16), retain the subset V_m and proceed to Step 7.

Step 7: Select another vertex not contained in V_m or V_m' and not previously tried and repeat Steps 4 through

Step 8 : When all the vertices in the complement of V_m' have been tried, define the resulting subset V_m'' as new V_1 and repeat Steps 7 through 9. Call each such complete repetition a cycle.

Step 9 : When one complete cycle of Steps 3 through 8 results in no reduction in h, terminate the procedure. The final V_m'' is all One optimal estimate of an m-vertex median of G.

At first sight this procedure appears laborious but it is much superior to Maranzana's method in many aspects and that will be quite evident in the Chapter V of this dissertation.

2.2 The Modified m-Vertex Median[10]

As mentioned in the Chapter I, this problem is nothing but the first problem with n new constraints, one for each vertex in the graph G.

If s_j is the limit for the vertex v_j i.e. the maximum weighted distance allowed that sepearates v_j from its nearest source vertex, then set

$$N_j = \{ v_i \mid h_{ij} \leq s_j \} \qquad (2.17)$$

$$\text{for } (j = 1,2, \ldots n)$$

will be the set of vertices within s_j of v_j and can provide acceptable service to the vertex v_j. For n vertices there will be n such sets and no set will be empty as h_{jj} is always zero while s_j is a positive quantity.

Here one may notice that every subset containing m vertices V_m of V may not be a feasible solution set to the problem as in the first problem. It may also happen, there exists no feasible solution set to the problem.

Any solution set V_m^p will be feasible if and only if

$$N_j \cap V_m^p \neq \phi \qquad (2.18)$$

$$\text{for } (j = 1, 2, \ldots . n)$$

If it is not possible to get such a V_m^p which satisfies the Eqn. (2.18), the problem will have no feasible solution.

So far a satisfactory method for solving the problem is not available. In the following subsections, the existing heuristic methods to solve the first problem are modified to solve this new problem.

2.2.1 Modified Maranzana's method:

The starting solution taken for this algorithm should be feasible too. This can be obtained by solving the

following system:

$$\sum_{i \in \alpha_j} y_i \geq 1 \qquad (2.19)$$

for $(j = 1, 2, \ldots n)$

$$\sum_{i=1}^{n} y_i = m$$

where $y_i \in \{0, 1\}$ and $\alpha_j = \{i \mid v_i \in N_j\}$

If $(y_1^*, \ldots, y_n^*)$ is any feasible solution then the corresponding ~~the~~ feasible solution set $V_m^* = \{v_i \mid y_i^* = 1.\}$

There are many methods to solve system (2.19). One elegant way is pseudo-Boolean method.

The modified algorithm is given below:

<u>Algorithm</u>

Step 1: Get a feasible solution set V_m^p as described above. Let it contain m vertices $v_{y_1}, \ldots, v_{y_m}$.*

Step 2: Determine a corresponding partition of V, $P_{y_1}, P_{y_2}, \ldots, P_{y_m}$ by putting

$$P_{y_j} = v_k \mid h_{y_j k} \leq h_{y_i k} \text{ and } y_i \in N_k$$

Step 3: Determine a centre of gravity, C_{y_j} for each P_{y_j}

satisfying $h_{1j,k}$ $h_{\ell jk}$ $\leq s_k$ for $v_k \in P_{y_j}$ also.

*In the algorithm y_i is not bivalent variable but at subscript which can take values from 1 to n.

(where $v_{1_j} = C_{y_j}$)

Step 4: If $C_{y_j} = v_{y_j}$ for all j, computation is stopped and the current values of v_{y_j} and P_{y_j} constitute the desired solution, otherwise set $v_{y_j} = C_{y_j}$ and return to Step 2.

2.2.2 Modified One optimal or Vertex substitution method:

Here also we have to start with a feasible solution as described above by solving system (2.19). While looking for a substitute of a vertex we have to always check whether the resulting solution set is feasible or not and h_p for solution set V_m^p is calculated as follows:

$$h_p = \sum_{j=1}^{n} h_{kj}$$

where $h_{kj} \leq h_{ij}$, $v_i \in V_m^p$ and $v_i \in N_j$

Other computations are same as in One optimal method.

2.3 Finding Minimum Number of Emergency Service Facilities[10]

In this problem we have to minimize the total number of sources required to meet the distance standards for each of the destinations. Hakimi[2] using Boolean functions, sought the minimum number of sources that covered all the destinations

within the specified maximum distance for ~~that~~ every destination. The resulting method requires an enumeration of all feasible solutions and as the problem size grows, the effort of determining the minimum number of sources can be expected to grow rapidly.

The problem can be mathematically expressed as follows:

$$\text{Min} \quad \sum_{i=1}^{n} y_i$$

such that

$$\sum_{i \in \alpha_j} y_i \geq 1 \qquad (2.20)$$

$$\text{for } (j = 1, 2, \ldots, n)$$

where $y_i = \{0, 1\}$, α_j have same definition as in the second problem.

There are three other methods which seem most favoured (10), these are:

(i) Linear programming and cutting plane techniques;

(ii) Reduction techniques;

(iii) Implicit enumeration.

The fourth approach developed is described in Chapter IV.

CHAPTER III

BASIC CONCEPTS OF PSEUDO-BOOLEAN PROGRAMMING

In this chapter basic concepts of pseudo-Boolean programming are reviewed[3,4].

Definition 3.1:

Any real valued function with bivalent variables is called a pseudo-Boolean function. Thus f will be a pseudo-Boolean function if

$$f : B_2^n \rightarrow R$$

where R is the field of real numbers and $B_2 = \{ 0, 1 \}$. It is obivious that such a function can be written as a polynomial linear in each variable.

Thus a function $f(x_1, \ldots, x_i, \ldots, x_n)$ can be written linear in ith variable as follows:

$$f(x_1, \ldots, x_n) = x_i \; g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_n) + h(x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \qquad (3.1)$$

Definition 3.2 :

An equality of the form

$$f \; (x_1, \ldots, x_n) = 0 \qquad (3.2)$$

where f is a pseudo-Boolean function, is called a pseudo-Boolean equality.

If Eqn. (3.2) would have been an inequality then it will be called pseudo-Boolean inequality.

Definition 3.3:

A pseudo-Boolean programme is the problem of optimizing a pseudo-Boolean function, whose variables can be either unconstrained or subjected to constraints expressed by a system of pseudo-Boolean inequalities and/or equalities. The optimization consists in finding the optimal value of the objective function $f(x_1, \ldots, x_m)$, and possibly determining one or all the optimizing points.

3.1 Solutions of Linear Pseudo-Boolean Equalities

Let $$p_1y_1 + q_1\bar{y}_1 + p_2y_2 + q_2\bar{y}_2 + \ldots + p_ny_n + q_n\bar{y}_n = s \tag{3.3}$$

be the general form of pseudo-Boolean equality where p_i, q_i $(i = 1, \ldots, n)$ and s are given constants.* We can very well assume $p_i \neq q_i$ for all i (otherwise the term $p_iy_i + q_i\bar{y}_i$ will be simply a constant and can be brought to right hand side of the equation).

---

* $\bar{y}_i = (1 - y_i)$ i.e. complement of y_i.

We will do a little bit of transformation for removing complement terms in a non decreasing order.

Let us set

$$x_i = \begin{cases} y_i & \text{if } p_i > q_i \\ \bar{y}_i & \text{if } p_i < q_i \end{cases} \qquad (3.4)$$

then $p_i y_i + q_i \bar{y}_i = \begin{cases} (p_i - q_i)\, x_i + q_i & \text{for } p_i > q_i \\ (q_i - p_i)\, x_i + p_i & \text{for } p_i < q_i \end{cases}$

and hence equation (3.3) becomes

$$r_1 x_1 + r_2 x_2 + \dots + r_n x_n = t \qquad (3.6)$$

where $r_1, r_2, \dots, r_n$, t are constants, $r_i > 0$ for each i and after reindexing we can suppose that

$$r_1 \geq r_2 \geq \dots \geq r_n > 0 \qquad (3.7)$$

Table (3.1) discusses 8 mutually exclusive cases concerning Eqn. (3.6).

The solutions of the Eqn. (3.6) which is the canonical form of Eqn. (3.3) can be tracked down along the branches of tree in Fig. 3.1 accordingly. Most of the blind alleys can be avoided by use of Table (3.1).

TABLE 3.1

No.	Case	Conclusions
1	$t < 0$	No solutions
2	$t = 0$	The unique solution is $x_1 = x_2 = \ldots = x_n = 0$
3	$t > 0$ and $r_1 \geq \ldots \geq r_m > t \geq r_{m+1} \ldots \geq r_n$	The solutions (if any) satisfy $x_1 = \ldots = x_m = 0$ and $\sum_{j=m+1}^{n} r_j x_j = t$
4	$t > 0$ and $r_1 = \ldots = r_m = t > r_{m+1} \ldots \geq r_n$	α) For every $k = 1,2, \ldots, m : x_k=1$ $x_1 = \ldots = x_{k-1} = x_{k+1} = \ldots = x_n = 0$ is a solution. β) The other solutions (if any) satisfy $x_1 = \ldots x_m = 0$ and $\sum_{j=m+1}^{n} r_j x_j = t$
5	$t > 0$, $r_i < t$ ii $= 1, \ldots n)$ and $\sum_{i=1}^{n} r_i < t$	No solutions
6	$t > 0$, $r_i < t$ $(i=1\ldots n)$ and $\sum_{i=1}^{n} r_i = t$	The unique solution is $x_1 = x_2 \ldots = x_n = 1$
7	$t > 0$, $r_i<t$ $(i=1,\ldots n)$ $\sum_{i=1}^{n} r_i>t$ and $\sum_{i=2}^{n} r_i<t$	The solutions (if any) satisfy $x_1 = 1$ and $\sum_{j=2}^{n} r_j x_j = t-r_1$
8	$t>0$, $r_i<t$ $(i=1,\ldots n)$ $\sum_{i=1}^{n} r_i>t$ and $\sum_{j=2}^{n} r_j \geq t$	The solutions (if any) satisfy either $x_1=1$ and $\sum_{j=2}^{n} r_j x_j=t-r_1$ or $x_1=0$ and $\sum_{j=2}^{n} r_j x_j = t$

The above method will lead to all the solutions of the Eqn. (3.6) and from solutions of the Eqn. (3.6), the solutions of the Eqn. (3.1) can be found.

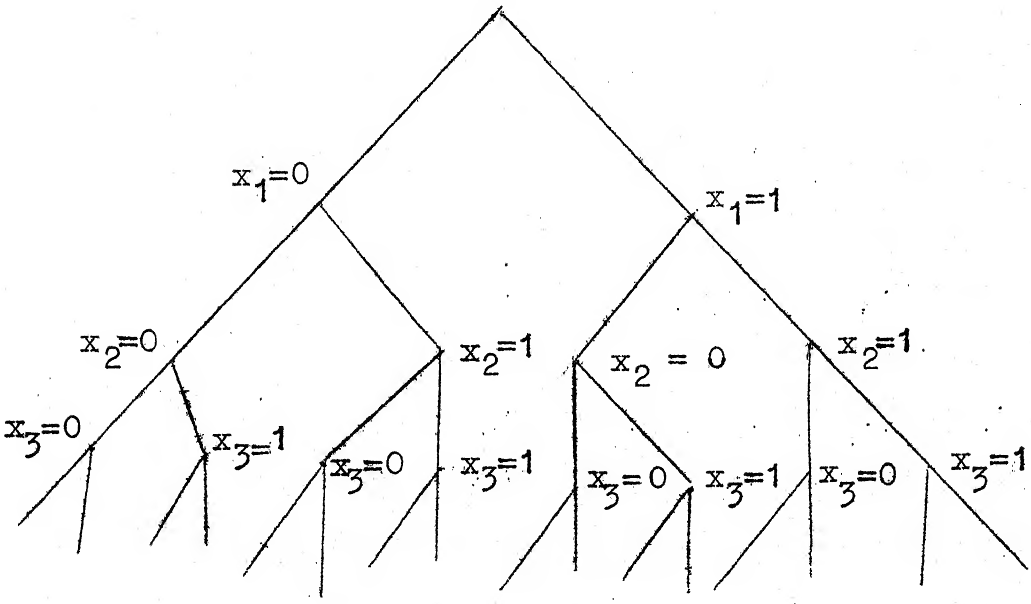


FIG. 3.1

3.2 Solutions of Linear Pseudo-Boolean Inequality

The most general form of a linear pseudo-Boolean inequality is either

$$p_1y_1 + q_1\bar{y}_1 + p_2y_2 + q_2\bar{y}_2 + \cdots + p_ny_n + q_n\bar{y}_n > r \quad (3.8)$$

or $$p_1y_1 + q_1\bar{y}_1 + p_2y_2 + q_2\bar{y}_2 + \cdots + p_ny_n + q_n\bar{y}_n \geq s \quad (3.9)$$

where p_i, q_i, r and s are constants, and we can assume $p_i \neq q_i$ as usual for all i. If p_i, q_i and r are integers, then Eqn.(3.8) can be written in the form of Eqn. (3.9) by taking s = r+1. In most of the cases p_i, q_i and r are integers, so we will focus our attention to in Eqn. (3.9). The method described below for solving Eqn. (3.9) will give solutionsof both

$$p_1y_1 + q_1\bar{y}_1 + \cdots + p_ny_n + q_n\bar{y}_n = s \quad (3.10)$$

$$p_1y_1 + q_1\bar{y}_1 + \cdots + p_ny_n + q_n\bar{y}_n > s \quad (3.11)$$

TABLE 3.2

No.	Case	Conclusions
1	$t \leq 0$	The unique basic solution is $x_1 = x_2 = \dots = x_n = 0$
2	$t > 0$ and $r_1 \geq \dots \geq r_m \geq t > r_{m+1} \geq \dots r_n$	α) For every $k = 1,2, \dots, m$: $x_k = 1$, $x_1 = \dots = x_{k-1} = x_{k+1} = \dots = x_n = 0$ is a basic solution. β) The other basic solutions (if any) are characterized by the property $x_1 = \dots = x_m = 0$, and $(x_{m+1}, \dots x_n)$ is a basic solution of $\sum_{j=m+1}^{n} r_j x_j \geq t$
3	$t > 0$, $r_i < t$ $(i=1, \dots n)$ and $\sum_{i=1}^{n} r_i < t$	No solutions
4	$t>0$, $r_i<t$ $(i=1,\dots n)$ and $\sum_{i=1}^{n} r_i = t$	The unique basic solution is $x_1 = x_2 = \dots = x_n = 1$
5	$t>0$, $r_i<t(i=1,\dots n)$ $\sum_{i=1}^{n} r_i>t$ and $\sum_{j=2}^{n} r_j<t$	The basic solutions (if any) are $x_1=1$, and $(x_2, \dots x_n)$ is a basic solution of $\sum_{j=2}^{n} r_j x_j \geq t - r_1$
6	$t>0$, $r_i<t(i=1,2,\dots n)$ $\sum_{i=1}^{n} r_i>t$ and $\sum_{j=2}^{n} r_j \geq t$	The basic solutions (if any) are characterized by the property: either $x_1=1$ and $(x_2, \dots x_n)$ is a basic solution of $\sum_{j=2}^{n} r_j x_j \geq t-r_1$ or $x_1=0$ and $(x_2, \dots, x_n)$ is a basic solution of $\sum_{j=2}^{n} r_j x_j \geq t$.

Definition 3.4:

Let $S = (y_1^*, \ldots, y_n^*)$ be a solution of Eqn. (3.9) and let I be a set of indices : $I \subseteq 1,2,\ldots,n$, Let (S,I) be set of all vectors $(y_1, \ldots, y_n) \in B_2^n$ satisfying

$$y_i = y_i^* \quad \text{for all } i \in I ,$$

the other variables y_j ($j \notin I$) being arbitrary. If all the vectors $(y_1, \ldots, y_n) \in \Sigma\ (S,I)$ satisfy the inequality (3.9) then $\Sigma\ (S,I)$ is said to be a family of solution S of Eqn.(3.9).

If $\#(I) < n$, the family is called nondegenerate.

We put inequality (3.9) into canonical form as in the previous case.

$$r_1x_1 + r_2x_2 + \ldots + r_nx_n \geq t \tag{3.12}$$

$$r_1 \geq r_2 \geq \ldots \geq r_n > 0 \tag{3.13}$$

A procedure given below enables to obtain the solutions of Eqn. (3.12) grouped into several nondegenerate and pairwise disjoint families of solutions, after this by applying inverse transformation we can obtain that of Eqn. (3.9).

Definition 3.5:

A vector $(x_1^*, \ldots, x_n^*)$ satisfying the inequality (3.12) is called a basic solution of (3.12), if for each index i such

that $x_i^* = 1$, the vector $(x_1^*, \ldots x_{i-1}^*, 0, x_{i+1}^*, \ldots, x_n^*)$ is not a solution of Eqn. (3.12).

Remark (3.1): The solutions of the eqation $r_1x_1 + \ldots + r_nx_n = t$ (if any) are also the basic solutions of Eqn. (3.12).

The solutions of Eqn. (3.12) are found by following two steps:

a) Determing all the basic solutions of Eqn. (3.12).

b) To each basic solution S_k, associating a certain set of indices I_k in such a way that $\Sigma\ (S_k, I_k)$ should be a family of solutions and that the system $\Sigma\ (S_k, I_k)_{k=1,\ldots m}$ should be 'complete' (i.e., it should include all solutions of Eqn. (3.12)).

First Step:

a) Determination of the basic solutions:

With the help of Table 3.2, all the basic solutions of (3.12) can be found as it was found with the help of Table 3. for Eqn. (3.6).

Second Step:

b) Determination of complete system of families of solutions of (3.12):

To each basic solution $S = (x_1^*, \ldots, x_n^*)$ we associate a family of solutions $\Sigma\ (S, J_S)$ as follows: Let i_o be the largest index for which $x_i^* = 1$, (i.e. $x_{i_o}^* = 1$ and $x_i^* = 0$ for all $i > i_o$) and let J_S be the set of all indices $i \leq i_o$. Then $\Sigma(S, J_S)$ is the set of all vectors $(x_1, \ldots, x_n)$ satisfying

$$x_i = \begin{cases} x_i^* & \text{for } i \le i_o \\ \text{arbitrary} & \text{for } i > i_o \end{cases} \qquad (3.14)$$

If $S_1, \ldots, S_m$ are all the basic solutions of Eqn. (3.12) and if $\sum_k (S_k, J_{S_k})$ $(k = 1, \ldots m)$ are constructed as above then these families will be all the solutions of canonical inequality (3.12).

Reverse transformation can be applied to obtain the solution of orignal inequality (3.9).

The solutions of Eqns. (3.15) and (3.16) can be seperated as

$$r_1 x_1 + \ldots + r_n x_n > t \qquad (3.15)$$

$$r_1 x_1 + \ldots + r_n x_n = t \qquad (3.16)$$

For Eqn. (3.16) it is immediate that the set of those solution of Eqn. (3.12) which satisfy equality (3.16) will give the complete system of solutions of Eqn. (3.16). (See Remark 3.1).

Let B be the set of all basic solutions of Eqn. (3.12). Let M' be the set of those basic solutions of Eqn. (3.12) which are not solutions of Eqn. (3.16). Let $S^* = (x_1^*, \ldots, x_n^*)$ be an element of B-M'. Let b be the largest index for which $x_b^* = 1$, we associate to S^*, the vectors $R_j^* = (z_{j1}^*, \ldots z_{jn}^*)$

for ($j = b+1, \ldots, n$) defined as follows:

$$z^*_{ji} = \begin{cases} x^*_i & \text{if } i \neq j \\ 1 = \bar{x}^*_j & \text{if } i = j \end{cases} \qquad (3.17)$$

the set of all vectors R^*_j for ($j = b+1, \ldots, n$) associated to the different elements of B-M' will be denoted by M" . Let M = M' U M" .

Now to find solutions of strict inequality (3.15) follow these steps:

a) Find M as described above.

b) Find the corresponding families of solutions.

Let $\Sigma(x_1, \ldots, x_n)$ denote a pseudo-Boolean equation or inequali

<u>Definition 3.6</u>:

The characteristic equation of $\Sigma(x_1, \ldots, x_n)$ is a Boolean equation

$$\phi\,(x_1, \ldots, x_n) = 1 \qquad (3.18)$$

which has the same solutions as $\Sigma(x_1, \ldots x_n)$; the Boolean function $\phi(x_1, \ldots, x_n)$ will be called <u>characteristic function</u> of $\Sigma(x_1, \ldots, x_n)$.

3.3 Determination of Characteristic Function in Various Cases

a) Linear equality:

$$\phi\ (x_1,\ \ldots,\ x_n) = \overset{\Sigma}{\underset{\alpha_1,\alpha_2,\ldots,\alpha_n}{U}} x_1^{\alpha_1} \ldots x_n^{\alpha_n} \ * \tag{3.19}$$

where $\overset{\Sigma}{\underset{\alpha_1,\ldots\alpha_n}{U}}$, means that union is extended over all the solutions $(\alpha_1,\ldots\alpha_n)$ of $\Sigma(x_1,\ldots x_n)$.

b) Linear inequalities:

$$\phi\ (x_1,\ldots x_n) = C_1\ U\ \ldots\ U\ C_p \tag{3.20}$$

where C_i is the Boolean function correspon-ding the to ith family solutions of linear inequality.

c) Nonlinear equality or inequality:

Let us consider a nonlinear pseudo-Boolean inequality with unknowns $x_1,\ldots x_n$.

$$p_1L_1 + \ldots + p_mL_m \geq q \tag{3.21}$$

where each L_i stands for a certain conjunction (i.e. a product of variables either complement or not):

$$L_i = x_{i_1}^{\theta_{i_1}} \ldots x_{i_{k(i)}}^{\theta_{i_{k(i)}}} \tag{3.22}$$

---

*Here $x_i^{\alpha_i} = x_i$ if $\alpha_i = 1$
$= \bar{x}_i$ if $\alpha_i = 0$

Let us replace the product by a single bivalent variable y_i and solve the resulting linear pseudo-Boolean **inequality**

$$p_1y_1 + \dots + p_my_m \geq q \tag{3.23}$$

where $y_1, \dots y_m$ are treated as independent variables. If $\psi\ (y_1, \dots y_m)$ is the characteristic function of the inequality (3.23) obtained as described in (b), then Boolean function

$$\phi\ (x_1, \dots x_n) = \psi\ \left[x_1^{\theta_{1_1}}, \dots x_{1_{k(1)}}^{\theta_{1_{k(1)}}}, \dots, x_{m_1}^{\theta_{m_1}}, \dots x_{m_{k(m)}}^{\theta_{m_{k(m)}}} \right] \tag{3.24}$$

is characteristic function of Eqn. (3.21). **Same procedure can** be applied for equa**li**ty.

3.4 Pseudo-Boolean form of Characteristic Function

We have seen that characteristic function of pseudo-Boolean system is a Boolean function. However, latter in Basic algorithm for minimizing pseudo-Boolean functions, it will be **necessary** to have a pseudo-Boolean expression of characteristic function i.e. an expression using only the arithmatical operations "+", '$\bar{-}$', and negation '$-$' of simple variables. These are the following identities for conversion:

$$a_1 U a_2 \ldots U a_n = \underbrace{(a_1+a_2+\ldots a_n)}_{n \text{ terms}} - \underbrace{(a_1a_2+a_1a_3+\ldots+a_1a_n+a_2a_3+\ldots a_2a_n}_{{}^nC_2 \text{ terms}}$$

$$\ldots.+a_{n-1}a_n) + \ldots \ \ldots\ldots\ldots$$

$$+ \underbrace{(-1)^{n-1} \quad a_1a_2a_3\ldots a_n}_{{}^nC_n = 1 \text{ term}} \tag{3.25}$$

If $\phi = C_1 \cup C_2 \ldots \cup C_n$, and if $C_iC_j = 0$ for all $i \neq j$

then

$$\phi = C_1 + C_2 + \ldots + C_n \tag{3.26}$$

3.5 Minimization of Unconstrained, Nonlinear Pseudo-Boolean Functions:

A vector $(x_1^*, \ldots, x_n^*) \in B_2^n$ is a minimizing point of the pseudo-Boolean function $f(x_1, \ldots x_n)$ if

$$f(x_1^*, \ldots x_n^*) \leq f(x_1, \ldots x_n) \tag{3.27}$$

for any $(x_1, \ldots x_n) \in B_2^n$, the value $f(x_1^*, \ldots x_n^*)$ is the minimum of f. From this definition it follows that

$$f(x_1^*, \ldots, x_n^*) \leq f(x_1^*, \ldots \bar{x}_i^*, x_{i+1}^*, \ldots x_n^* \qquad (3.28(1)$$

$$\vdots$$

$$\text{for all } i = 1,2,\ldots n \qquad (3.28($$

Conditions (3.28(1)) to (3.28(n)) are necessary but not suffic

for $(x_1^*, \ldots, x_n^*)$ to be a minimizing point but they characterize local minima of f.

Basic algorithm:

The algorithm is made up of two main stages. In first one, the minimum of the given pseudo-Boolean function $f(x_1, \ldots, x_n)$ is found, while in second all the minimizing points are determined.

First stage: Let us denote, for the sake of recurrence

$$f_1(x_1, \ldots, x_n) = f(x_1, \ldots, x_n)$$

f_1 can be expressed as

$$f_1(x_1, \ldots, x_n) = x_1 g_1(x_2, \ldots, x_n) + h_1(x_2, \ldots, x_n) \tag{3.29(1)}$$

Inequality (3.28(1)) becomes by using Eqn. (3.29(1))

$$(x_1 - \bar{x}_1) g_1(x_2, \ldots, x_n) \leq 0 \tag{3.30(1)}$$

If ψ_1' and ψ_1'' are the characteristic functions of the inequality $g_1 < 0$ and the equation $g_1 = 0$. x_1 can be written as

$$x_1 = \psi_1 (p_1, x_2, \ldots, x_n) = \psi_1' (x_2, \ldots, x_n) \cup p_1 \psi_1'' (x_2, \ldots, x_n) \tag{3.31(1)}$$

where p_1 is an arbitrary bivalent variable.

This completes first step.

At second step

$$f_2(x_2,\ldots,x_n) = f_1(\psi_1'(x_2, \ldots,x_n), x_2,\ldots,x_n) \tag{3.32(2)}$$

Since f_2 is a pseudo-Boolean function, ψ_1' should be in pseudo-Boolean form, Hence

$$f_2(x_2,\ldots,x_n) = \psi_1' \ (x_2,\ldots,x_n) \ g_1(x_2,\ldots,x_n) + h_1(x_2,\ldots,x$$

Continuing this way, we obtain at nth step a function $f_n(x_n)$ which is written in the form

$$f_n(x_n) = x_n g_n + h_n \tag{3.29(n)}$$

where g_n and h_n are constants.

We have to solve inequality

$$(x_n - \bar{x}_n) \ g_n \leq 0$$

The solution is

$$x_n = \psi_n \ (p_n)$$

where $\psi_n(p_n) = \begin{cases} 1 & \text{if } g_n < 0 \\ 0 & \text{if } g_n > 0 \\ p_n & \text{if } g_n = 0 \end{cases}$

The minimum of orignal function f is

$$f_{min} = f_{n+1} = f_n\ (\psi_n')$$

where the constant ψ_n' is given by

$$\psi_n' = \begin{cases} 1 & \text{if } g_n < 0 \\ 0 & \text{if } g_n \geq 0 \end{cases}$$

The first stage comes to an end.

Second stage: The various minimizing points can be determined by recursion as follows:

$$x_n = \psi_n\ (p_n)$$

$$x_{n-1} = \psi_{n-1}\ (p_{n-1},\ \psi_n\ (p_n))$$

$$\vdots$$

$$x_1 = \psi_1\ (p_1,\ \ldots,\ p_{n-1},\ p_n)$$

Recently many new methods for minimizing the pseudo-Boolean functions have been suggested[5,6].

CHAPTER IV

PSEUDO-BOOLEAN FORMULATION

This chapter is devoted to the development of the methods for getting the optimal results for all the three problems. This has been done by reducing the task to that of minimization of unconstrained nonlinear pseudo-Boolean functions. Proofs have also been given to ensure that minimization of these unconstrained nonlinear pseudo-Boolean functions will give the optimal results.

4.1 The m-Vertex Median of a Graph:

Let us recall of our notations first:

G : Given grapth with n vertices;

D : Distance matrix (n x n), d_{ij} represents the minimum distance between vertex v_i and v_j.

X : nth order diagonal matrix with vertex weights on the diagonal.

H : Weighted distance matrix, h_{ij} representing minimum weighted distance of vertex v_j from v_i.

$$H = D\,X = \left| h_{ij} \right| = \left| d_{ij}\, x_j \right|$$

We have to find a source subset V_m^{p*} ($\#(V_m^{p*}) = m$) such that

$$h_{p*} \leq h_p$$
$$\text{for } p \in 1, 2, \ldots \binom{n}{m}$$

where
$$h_p = \sum_{j=1}^{n} \min_{v_i \in V_m^p} (h_{ij}) \tag{4.1}$$

Define the following variables:

y_i = 1, if a source is located at vertex v_i
= 0, otherwise

t_{ij} = 1, if there is a source at v_i and v_j served by v_i.
= 0, otherwise.

This problem (say P) can be written as follows:

Problem P

$$\text{Min} \sum_{j=1}^{n} \sum_{i=1}^{n} t_{ij} h_{ij} \tag{4.2}$$

$$\sum_{i=1}^{n} y_i = m \tag{4.3}$$

$$\sum_{i=1}^{n} t_{ij} = 1 \tag{4.4}$$
(for all $j=1,\ldots,n$)

and $y_i = 0 \Longrightarrow t_{ij} = 0$ for all $j = 1,2, \ldots, n$ (4.5)

One can easily make out that t_{ij} are dependent variables as for a given Y vector of y_i's, the corresponding T matrices of t_{ij} 's can be easily determined.

Let us define

$$v_{ikj} = \begin{cases} 0 & \text{if } h_{ij} < h_{kj} \\ 0 & \text{if } h_{ij} = h_{kj} \text{ and } i \leq k \\ 1 & \text{if } h_{ij} = h_{kj} \text{ and } i > k \\ 1 & \text{if } h_{ij} > h_{kj} \end{cases} \quad (4.6)$$

Express t_{ij} as follows:

$$t_{ij} = y_i \prod_{k=1}^{n} (1 - y_k v_{ikj}) \quad (4.7)$$

Let us consider the following problem:

Problem P'

$$\text{Min. } \sum_{j=1}^{n} \sum_{i=1}^{n} \left[y_i \prod_{k=1}^{n} (1 - y_k v_{ikj}) \right] h_{ij} \quad (4.8)$$

$$\sum_{i=1}^{n} y_i = m \quad (4.9)$$

<u>Theorem 4.1</u>: The minimum of Problem P is equal to the minimum of Problem P'. If $(y_1^*, \ldots, y_n^*)$ is an optimal solution to problem P' and if t_{ij}^* the corresponding values of t_{ij} (by expression 4.7), $(y_1^*, \ldots y_n^*, t_{11}^*, \ldots, t_{nn}^*)$ is an optimal solution to P.

The proof of the theorem will be followed by the proofs of the following lemmas:

Lemma 1: If $S = (y_1, \ldots, y_n)$ is a feasible solution of the Problem P' then the corresponding solution $S' = (y_1, \ldots, y_n, t_{11}, \ldots, t_{nn})$ will be a feasible solution to Problem P (t_{ij} 's are determined according to expression(4.7)).

Proof: We will prove that S' satisfies all constraints of the Problem P thus making it feasible.

S' will satisfy constraint (4.3) as S satisfies constraint (4.9).

t_{ij} in S' are expressed as follows:

$$t_{ij} = y_i \prod_{k=1}^{n} (1 - y_k v_{ikj})$$

In the above expression if $y_i = 0$, then $t_{ij} = 0$ for $j = 1, \ldots, n$. Therefore, solution S' satisfies the constraint set (4.5) too.

Now we have to show that S' satisfies the constraint set (4.4) too.

Let us define $I_L = \{ k \mid y_k = 1 \}$

From above expression for t_{ij} and definition of v_{ikj} according to (4.6) we will prove below that for each j, there exists one and only one s_j such that t_{sj}, j = 1 and for that

s_j, the following are true:

$$y_{s_j} = 1$$

and

either $\quad h_{s_j j} < h_{kj} \qquad$ for $k \in I_L \qquad (4.10)$

or

$$h_{s_j j} = h_{kj} \text{ and } s_j \leq k$$

We have to first show that if system (4.10) is satisfied then $t_{s_j j} = 1$

$$t_{s_j j} = y_{s_j} \prod_{k=1}^{n} (1 - y_k v_{s_j kj})$$

$$= \prod_{k \in I_L} (1 - v_{s_j kj})$$

$$v_{s_j kj} = 0 \qquad \text{for } k \in I_L$$

as either $h_{s_j j} \leq h_{kj}$

or $\quad h_{s_j j} = h_{kj}$ and $s_j \leq k$

Hence $t_{s_j j} = 1$

For any j, there always exists one s_j satisfying system (4.10).

We will prove that for every j

$$t_{pj} = 0 \quad \text{for} \quad p \neq s_j$$

If $p \notin I_L$ then $y_p = 0$ and subsequently $t_{pj} = 0$. So we have to prove

$$t_{pj} = 0 \quad \text{for} \quad p \in I_L \text{ and } p \neq s_j$$

From system (4.10)

$$\begin{aligned} &\text{either} \quad h_{pj} > h_{s_j j} \\ &\text{or} \qquad h_{pj} = h_{s_j j} \quad \text{and } p > s_j \end{aligned} \qquad (4.11)$$

System (4.11) says that $v_{p\, s_j} = 1$

$$\begin{aligned} t_{pj} &= \prod_{k \in I_L} (1 - v_{p\,k\,j}) \quad \text{for all } p \in I_L \text{ and } p \neq s_j \\ &= 0 \\ &\qquad\qquad (\text{as for } k = s_j,\ v_{ps_j} = 1) \end{aligned}$$

Therefore

$$\begin{aligned} \sum_{p=1}^{n} t_{pj} &= \sum_{p \notin I_L} t_{pj} + \sum_{\substack{p \in I_L \\ p \neq s_j}} t_{pj} + t_{s_j j} \\ &= t_{s_j, j} \\ &= 1 \end{aligned} \qquad (4.12)$$

Now Eqn. (4.12) is true for every j, therefore it shows that S' satisfies the constraint set (4.4) too.

Hence S' satisfies all the constraints of Problem P, thus proving lemma 4.1.

Let $(t^*_{11}, \ldots, t^*_{nn}, y^*_1, \ldots, y^*_n)$ be any feasible solution to Problem P and let f^* denote the value of the objective function of P at this point.

Let us put

$$t^{**}_{ij} = y^*_i \prod_{k=1}^{n} (1 - y^*_k v_{ijj})$$

then $(t^{**}_{11}, \ldots, t^{**}_{nn}, y^*_1, \ldots, y^*_n)$ is also a feasible solution to P (By lemma 4.1). Let f^{**} denote the value of objective function of P at this point.

<u>Lemma 4.2</u>: $f^{**} \leq f^*$

Proof:
$$f^* = \sum_{j=1}^{n} \sum_{i=1}^{n} h_{ij} t^*_{ij}$$

$$= \sum_{j=1}^{n} h_{s_j j} + \sum_{j=1}^{n} \sum_{i=1}^{n} (h_{ij} - h_{s_j j}) t^*_{ij}$$

$$\geq \sum_{j=1}^{n} h_{s_j j} = f^{**}$$

Hence lemma 4.2 is true.

It follows directly from lemma 4.1 and lemma 4.2 that the minimum of problem P' is also the minimum of problem P.

Thus we have reduced P to P'.

Let $V = \sum_{j=1}^{n} \max_{i=1,2,\ldots,n} (h_{ij})$

Lemma 4.3: Any solution to the problem P' or P will never give the value of the objective function more than V.

Proof:- Obivious.

Consider problem P"

Problem P" $\quad$ Min $\sum_{j=1}^{n} \sum_{i=1}^{n} y_i \prod_{k=1}^{n} (1-y_k v_{ikj}) h_{ij} + (V+1)(\sum_{i=1}^{n} y_i - m$

(4.13)

Theorem 4.2: The minimum of the problem P" will also be the minimum of problem P'.

Proof of this theorem will follow the proof of following lemma:

Lemma 4.4: Any optimal solution S which gives minimum value of (4.13) (problem P") satisfies $\sum_{i=1}^{n} y_i = m$.

Proof: If $\sum_{i=1}^{n} y_i \neq m$, then value of (4.13) will be greater than V contradicting the fact that S gives minimum value of (4.13); because any solution satisfying $\sum_{i=1}^{n} y_i = m$ in (4.13)

will always give the value of (4.13) not exceeding V.

Hence lemma is true.

The proof of the theorem directly follows from lemma 4.4, because if $\sum_{i=1}^{n} y_i = m$ then (4.13) and (4.8) becomes same; satisfying constraint (4.9) too. Therefore the minimum of P" will be the minimum of P' too.

It follows directly from Theorems (4.1) and (4.2) that the minimum of P" will be the minimum of P too. Thus we have reduced the problem to minimization of an unconstrain nonlinear pseudo-Boolean function.

Hammer[3,4] has proposed Basic algorithm to minimize these unconstrained, nonlinear pseudo-Boolean functions.

In Chapter V, some important points are mentioned to code this algorithm efficiently on computer.

Some other ideas for minimizing these types of functions have also been proposed recently[5,6].

Formulation and proofs for the third problem will help in better understanding of formulation and proofs for the second problem, therefore the third problem follows first

4.2 Finding Minimum Number of Emergency Service[10]

Let us look again at the problem, we have

s_j : the maximum weighted distance allowed that seperates

vertex v_j from its nearest source vertex.

We have to find the minimum number of sources required to meet these distance standards with other things as usual.

If we define p_{ij} as

$$p_{ij} = 0 \quad \text{if } h_{ij} > s_j$$
$$\quad = 1 \quad \text{if } h_{ij} \leq s_j$$

then this problem can be expressed mathematically as follows

Problem Q

$$\text{Min} \sum_{i=1}^{n} y_i \tag{4.14}$$

$$\sum_{i=1}^{n} p_{ij}\, y_i \geq 1 \quad \text{for}(j = 1,2, \ldots, n) \tag{4.15}$$

Consider the following problem Q'

Problem Q'

$$\text{Min} \sum_{i=1}^{n} y_i + (n+1) \sum_{j=1}^{n} \sum_{i=1}^{n} (\overline{p_{ij}\, y_i}) \tag{4.16}$$

<u>Theorem 4.3</u>: The minimum for the problem Q' is also the minimum for problem Q.

The proof of the above theorem will follow from the following lemmas:

Lemma 4.5: The minima for both Q and Q' will never be greater than n.

Proof: Taking solution S with all y_i's equal to one, we observe the following:

(i) Solution S is feasible to problem Q, as the constraint set (4.15) is satisfied, as $p_{jj} = 1$ for $(j = 1,\ldots,n)$.

(ii) The value of the objective function of Q corresponding to solution S is n.

(iii) Solution S is feasible to Q', as Q' does not have any constraint.

(iv) The value of objective function of Q' correspondir to solution S is n.

The proof of the lemma follows directly from (i), (ii), (iii), (iv).

Lemma 4.6: Any optimal solution S of problem Q' satisfies

$$\prod_{i=1}^{n} (\overline{p_{ij}\ y_i}) = 0 \qquad (4.17)$$

Proof: If system (4.17) is not satisfied for any of j then the objective function (4.16) will be greater than n and

hence contradicting the fact that S gives minimum of Q'.

Lemma 4.7: Any optimal solution S of the problem Q' will satisfy the constraint set (4.15) i.e.

$$\sum_{i=1}^{n} p_{ij}\, y_i \geq 1 \qquad \text{for } (j = 1,2,\ldots,n)$$

Proof: By lemma 4.6, S will satisfy

$$\prod_{i=1}^{n} (\overline{p_{ij}\, y_i}) = 0 \qquad \text{for } (j = 1,\ldots,n)$$

This implies for every j, there should exist atleast one i, such that

$$p_{ij}\, y_i = 1$$

i.e. $$\sum_{i=1}^{n} p_{ij}\, y_i \geq 1$$

Therefore S satisfies $$\sum_{i=1}^{n} p_{ij}\, y_i \geq 1 \qquad \text{for } (j = 1,2,\ldots,n)$$

Lemma 4.8: Any solution, satisfying the constraint set (4.15) i.e.

$$\sum_{i=1}^{n} p_{ij}\, y_i \geq 1 \qquad \text{for } (j = 1,2,\ldots,n),$$

satisfies $\prod_{i=1}^{n} (\overline{p_{ij}\ y_i}) = 0$

for $(j = 1,2,\ldots,n)$

Proof: Obivious by observation.

It follows from the above lemma that for all feasible solutions to problem Q, the problem Q' is

$$\text{Min} \sum_{i=1}^{n} y_i$$

and lemma 4.6 says minimum of Q' will always satisfy

$$\sum_{j=1}^{n} \prod_{i=1}^{n} (\overline{p_{ij}\ y_j}) = 0$$

From the above two statements and lemma 4.7, it follows directly that the minimum of Q' is also the minimum of Q.

Thus once again we have reduced this problem to the minimization of an unconstrained, nonlinear pseudo-Boolean function.

The second problem follows:

4.3 The Modified m-Vertex Median of a Graph[10]

As already mentioned in 2.2, in this modified m-vertex median problem we have to satisfy the distance constraints of the third problem too, when solving the first problem, m-vertex median.

The problem can be expressed mathematically as follows:

Problem R

$$\text{Min} \sum_{j=1}^{n} \sum_{i=1}^{n} y_i \prod_{k=1}^{n} (1-y_k\, v_{ikj})\; h_{ij} + (V+1) \left(\sum_{i=1}^{n} y_i - m \right)^2 \qquad (4.18)$$

such that

$$\sum_{i=1}^{n} y_i\, p_{ij} \geq 1 \quad \text{for } (j = 1,2,\ldots,n) \qquad (4.19)$$

It may happen that there exists no feasible solution to modified m-vertex median of the given graph G. This depends on s_j's and m for a given H for the graph G. Now even if there does not exist a feasible solution to m-vertex median of G, there will exist a feasible solution to R, but minimum of R will be larger than V. We already know minimum of R will never be larger than V if there exists a feasible solution to modified m-vertex median.

Hence, this information should be kept in mind; if minimum of R happens to be more than V, then there does not exist a feasible solution to the modified m-vertex median.

Consider problem R' as follows

Problem R' $\quad$ Min $\sum_{j=1}^{n} \sum_{i=1}^{n} y_i \prod_{k=1}^{n} (1-y_k v_{ikj}) h_{ij} + (V+1)$

$$\left(\sum_{i=1}^{n} y_i - m\right)^2 + (V+1) \sum_{j=1}^{n} \sum_{i=1}^{n} (\overline{y_i p_{ij}})$$

<u>Theorem 4.3</u>: If there exists a feasible solution to the modified m-vertex median of the given graph G, then the minimum of R' will be the minimum of R.

The proof will follow the proof of the lemma given below.

<u>Lemma 4.9</u>: If there exists a feasible solution to the modified m-vertex median of the given graph G, then a solution S giving minimum of R' will satisfy the constraint set (4.19) of R.

Proof: Now if there exists a feasible solution to modified m-vertex median of G, then the minimum of R' will not be greater than V, hence S will satisfy

$$\prod_{i=1}^{n} (\overline{y_i p_{ij}}) = 0 \qquad (4.20)$$

for $(j = 1,\ldots,n)$

system (4.20) shows

$$\sum_{i=1}^{n} y_i p_{ij} \geq 1$$

for $(j = 1, \ldots, n)$

Hence S satisfies constraint set (4.19) of R i.e. it is a feasible solution of R.

This proves lemma 4.9.

<u>Lemma 4.10</u>: Any feasible solution S of R will satisfy

$$\sum_{j=1}^{n} \prod_{i=1}^{n} (\overline{y_i \, p_{ij}}) = 0$$

Since

$$\sum_{i=1}^{n} y_i \, p_{ij} \geq 1 \qquad \text{for } (j = 1, \ldots, n),$$

$$\prod_{i=1}^{n} (\overline{y_i \, p_{ij}}) = 0 \qquad \text{for } (j = 1, \ldots, n)$$

Hence

$$\sum_{j=1}^{n} \prod_{i=1}^{n} (\overline{y_i \, p_{ij}}) = 0$$

This proves lemma 4.10.

The proof of theorem 4.3 follows directly from proof of lemmas 4.9 and 4.10.

Here also we have reduced the problem to minimization of unconstrained, nonlinear pseudo-Boolean function.

While we minimize these unconstrained, nonlinear pseudo-Boolean functions we get not only single optimal solution for the problem, but also all the optimal solutions existing for that problem.

CHAPTER V

COMPUTATIONAL ATTRIBUTES

In this chapter principles of computer code for Basic algorithm to find all m-vertex medians of a graph are described. Also the computational attributes of various heuristic algorithms for the above problem are compared. Summary of the results obtained on IBM/7044 for problems of various sizes and the conclusions made for them are also given.

5.1 Coding of Basic Algorithm

In Chapter III, Basic algorithm to minimize unconstrained pseudo-Boolean functions was described. In brief, it contained essentially the following:

Let $f(y_1,\ldots,y_n)$ be the function to be minimized. Consider the following system

$$f(y_1,\ldots,y_n) \leq f(y_1,\ldots,y_{i-1}, \bar{y}_i, y_{i+1},\ldots,y_n) \qquad (5.1(1))$$

$$\text{for } (i = 1,2,\ldots,n) \qquad \vdots \quad (5.1(n))$$

If we successively solve the inequalities (5.1(1)) to (5.1(n)) introducing in each inequality the parametric solution of the preceding one, we will obtain all the minimizing points.

There are essentially two stages. In first stage we find minimum of f and in the second stage all the minimizing points of f.

First stage: For the sake of recurrence we write

$$f_1(y_1, \ldots, y_n) = f(y_1,\ldots,y_n) \tag{5.2}$$

and express $f_1(y_1,\ldots,y_n) \quad = y_1 g_1(y_2,\ldots,y_n) + h_1(y_2,\ldots,y_n)$

$$\tag{5.3}$$

We find ψ_1' and ψ_1'' the characteristic functions of $g_1 < 0$ and $g_1 = 0$ and then we get f_2 as follows:

$$f_2(y_2, \ldots,y_n) = f_1(\psi_1'(y_2,\ldots,y_n), y_2,\ldots,y_n) \tag{5.4}$$

By converting ψ_1' in pseudo-Boolean form

$$f_2(y_2,\ldots,y_n) = \psi_1' (y_2,\ldots,y_n) \; g_1(y_2,\ldots,y_n) + h_1(y_2,\ldots,y_n)$$

We store the value of y_1 in terms of $(y_2,\ldots,y_n)$ given by the following expression (5.5), which will be used in the second stage to determine all the minimizing points of f.

$$y_1 = \psi_1(p_1, y_2,\ldots,y_n) = \psi_1'(y_2,\ldots,y_n) \cup p_1 \; \psi_1''(y_2,\ldots,y_n)$$

$$\tag{5.5}$$

where p_1 is an arbitrary bivalent variable. This completes

first step.

We proceed now with the function f_2 in the same way as with f_1. Continuing this way after nth step we have

$$f_{min} = f_{n+1} = f_n\ (\psi_n') \tag{5.6}$$

and

$$y_n = \psi_n\ (p_n) \tag{5.7}$$

where p_n is an arbitrary bivalent variable.

The first stage comes to an end.

Second stage: Introducing the values of y_n (obtained above) in the expression for y_{n-1} we obtain the corresponding values of y_{n-1} and then introducing values of y_n and y_{n-1} in the expression for y_{n-2} we get the values of y_{n-2}. Continuing in this way we obtain all the minimizing points of f.

Considering the problem of m-vertex median, we have to minimize the following unconstrained, nonlinear pseudo-Boolean function:

$$f(y_1,\ldots,y_n) = \sum_{j=1}^{n} \sum_{i=1}^{n} y_i \prod_{k=1}^{n} (1-y_k\ v_{ikj})\ h_{ij} + (V+1)\left(\sum_{i=1}^{n} y_i - m\right)^2 \tag{5.8}$$

The number of terms involved and number of elements in the terms for expression (5.8) are highly undeterministic, depending upon the problem. Using 'SLIP'* routines for coding Basic algorithm will be an elegant way to minimize above function.

One way to solve this problem is to generate the whole expression (5.8) from given H matrix, store it in the core memory and then apply Basic algorithm as stated above. In this way the memory requirement will be too large limiting the size of problem which can be solved on a digital computer.

We observed in the description of Basic algorithm that at any step i in the first stage we require only g_i for getting ψ_i', ψ_i'' and ψ_i. This observation suggests an elegant way of utilization of memory by not putting the whole expression (5.8) in the core to start with, but generating it in parts as and when the need arises. By this procedure we will be minimizing the following function f^*:

$$f^* = f - (V+1)\, m^2 \qquad (5.4)$$

Since $(V+1)\, m^2$ is a constant term, the minimizing points of f^* will be same as that of f and

$$f_{min} = f^*_{min} + (V+1)\, m^2 \qquad (5.9)$$

---

*'SLIP' routines for list processing are available on IBM/7044 at I.I.T. Kanpur, India.

At first step we will generate all those terms which have y_1 as an element. Let us denote if as f_1'. f_1' is given by the following expression*

$$f_1' = y_1 \sum_{j=1}^{n} h_{1j} \prod_{k=2}^{n} (1-y_k v_{1kj}) + (V+1)y_1 \left[(1-2m) + 2 \sum_{s=2}^{m} y_s\right]$$

$$- y_1 \sum_{j=1}^{n} \sum_{i=2}^{n} h_{ij} y_i v_{i1j} \prod_{k=2}^{n} (1-y_k v_{ikj}) \qquad (5.10)$$

We will make $f_1 = f_1'$ and find g_1 by Eqn. (5.3). Subsequently we find ψ_1', ψ_1'' and ψ from this g_1 as usual.

We find $f_2''(y_2,\ldots,y_n)$ as follows:

$$f_2''(y_2,\ldots,y_n) = f_1(\psi_1', (y_2,\ldots,y_n), y_2,\ldots,y_n) \qquad (5.11)$$

Now we generate f_2' by following expression**.

$$f_2'(y_2,\ldots,y_n) = y_2 \sum_{j=1}^{n} h_{2j} \prod_{k=3}^{n} (1-y_k v_{2kj})$$

$$- y_2 \sum_{j=1}^{n} \sum_{i=3}^{n} h_{ij} y_i v_{i2j} \prod_{k=3}^{n} (1-y_k v_{ikj})$$

$$+ (V+1) y_2 \left[(1-2m) + 2 \sum_{s=3}^{n} y_s\right] \qquad (5.12)$$

---

* The expression (5.11) can be obtained by putting r = 1 in expression (5.13).

** The expression (5.12) can be obtained by putting r = 2 in expression (5.13).

Let us construct the function f_2 as follows:

$$f_2 = f_2' + f_2''$$

We proceed now with f_2 in the same way as with f_1. At step r we will find

$$f_r = f_r' + f_r''$$

where

$$\begin{aligned} f_r'(y_r,\ldots,y_n) = {} & y_r \sum_{j=1}^{n} h_{rj} \prod_{k=r+1}^{n} (1-y_k\, v_{rkj}) \\ & - y_r \sum_{j=1}^{n} \sum_{i=r+1}^{n} h_{ij}\, y_i\, v_{irj} \prod_{k=r+1}^{n} (1-y_k\, v \\ & + (V+1)\, y_r \left[(1-2m) + 2 \sum_{s=r+1}^{n} y_s \right] \end{aligned}$$

(5.13)

Continuing in this way after nth step we have

$$f^*_{min} = f_{n+1} = f''_{n+1} = f_n\,(\psi_n') \qquad (5.14)$$

and

$$y_n = \psi_n\,(p_n) \qquad (5.15)$$

where p_n is an arbitrary bivalent variable.

Now

$$f_{min} = f^*_{min} + (V+1)m^2$$

After this, all the minimizing points of f will be determined as usual.

These observations direct us to develop an algorithm as follows:

Step 1: Generate f'_1 by expression (5.10) and make $f_1 \leftarrow f'_1$.

Step 2: Find g_1 and from that corresponding Ψ'_1, Ψ''_1, Ψ_1 and f''_2. Save Ψ_1.

Step 3: $r \leftarrow 2$

Step 4: Generate f'_r by expression (5.13).

Step 5: $f_r \leftarrow f'_r + f''_r$

Step 6: Find corresponding g_r from f_r. And from g_r the corresponding Ψ'_r, Ψ''_r, Ψ_r and f''_{r+1}. Save Ψ_r.

Step 7: If $r < n$ then $r \leftarrow r+1$ and go to Step 4. If not go to Step 8.

Step 8: $f^*_{min} \leftarrow f''_{n+1}$

Step 9: $f_{min} \leftarrow f^*_{min} + (V+1)m^2$

Step 10: Determine all the minimizing points from $\psi_1, \psi_2, \ldots, \psi_n$ as described in Stage 2 of the Basic algorithm.

At Step 9, we get the minimum value of objective function i.e. f_{min} and at Step 10, we get all the minimizing point of f.

A similar approach can be adopted for the second problem i.e. modified m-vertex median and the third problem i.e. finding minimum number of emergency services.

A code for the algorithm described above was prepared using 'SLIP' routines available on Computer IBM/7044. This code has mainly three routines whose functions are given below:

(i) Routine OPTAL:- This routine creates list structures IPRO, ISTO, IVAR and ISVAR. The lists ISTO AND IPRO are used to store f_r in parts at any step r. List IPRO contains expression for g_r (one sublist of IPRO representing one term of g_r) before it calls the Routine BALGO. The Routine BALGO and subsequently Routine ALSOLN do necessary things to store ψ_r' and ψ_r'' in lists ISVAR and IVAR respectively, and also update IPRO. Now we increase index r to r+1 and repeat the whole procedure. When value of r becomes n this Routine OPTAL finds f_{min}, i.e. the minimum value of function

f, and all values of variable y_n. All the minimizing points of f are now found by processing lists ISVAR and IVAR.

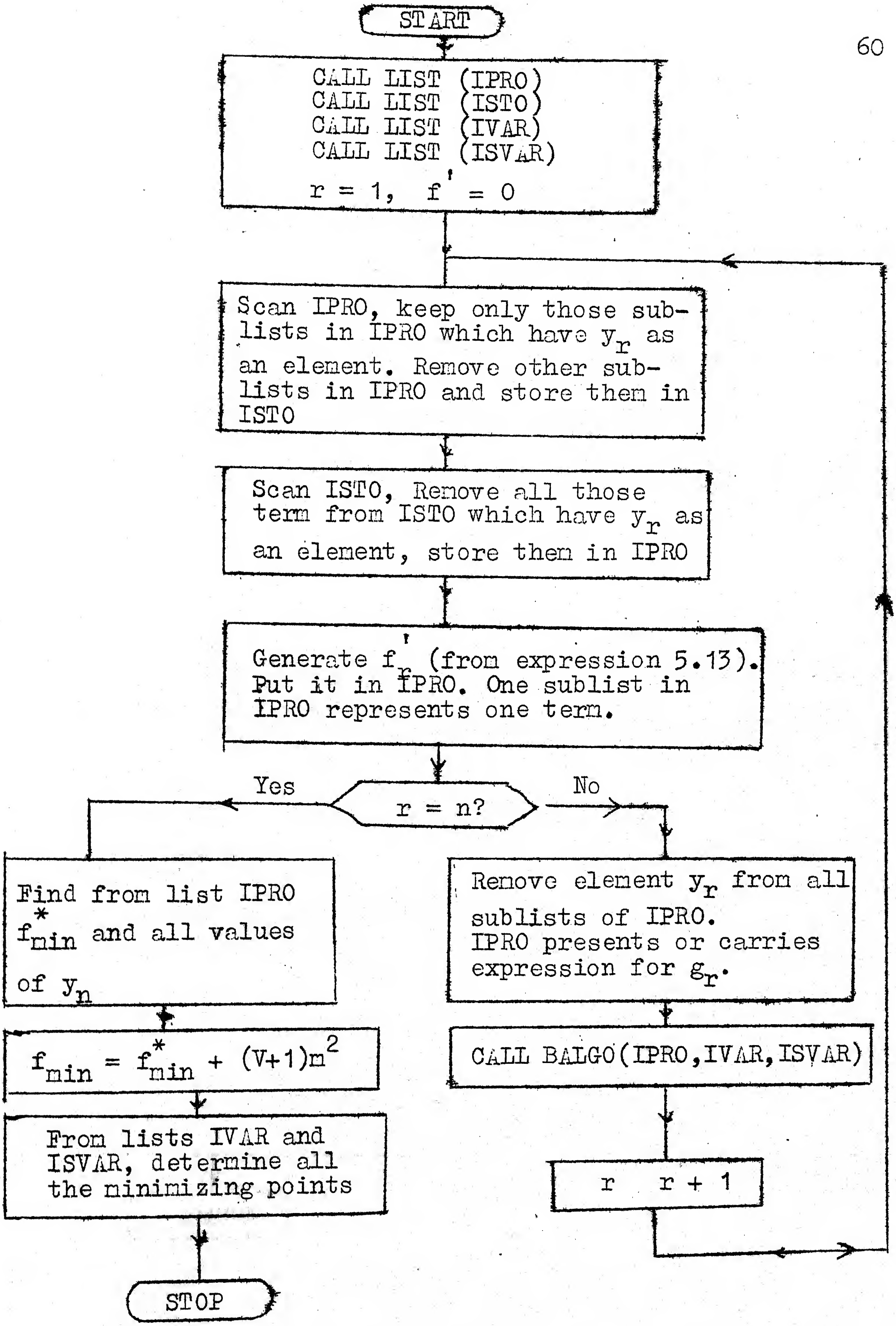
(ii) Routine BALGO: It processes list IPRO to get all the solutions of $g_r < 0$ and $g_r = 0$, and stores them in list ISOLN. Out of all sublists of ISOLN, those sublists which have a mark correspond to the solutions of $g_r = 0$.

After this it calls Routine ALSOLN. Finally it updates IPRO and returns back.

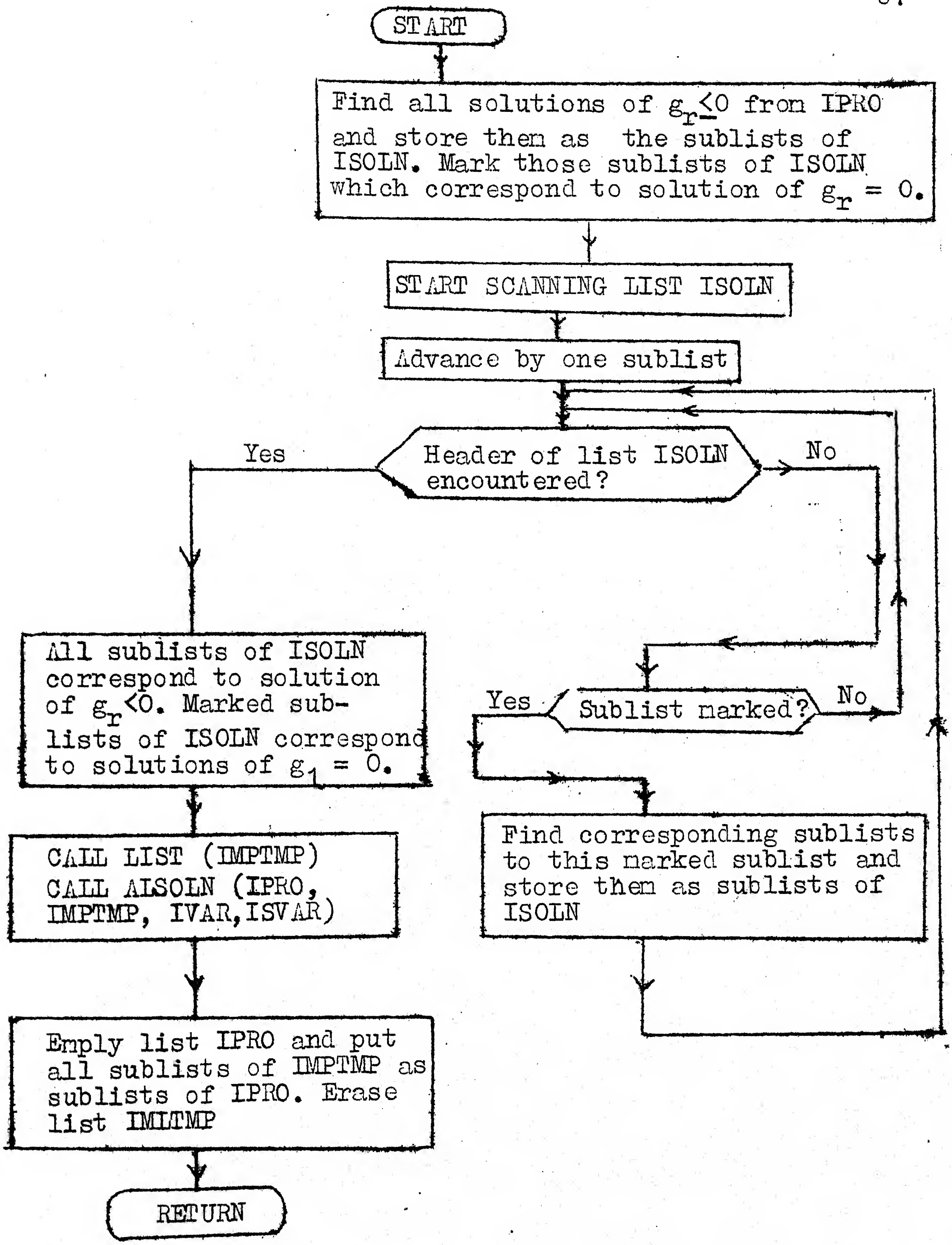
(iii) Routine ALSOLN: This routine finds ψ_r' and ψ_r'' from list ISOLN and stores them in lists ISVAR and IVAR respectively. ψ_r' is converted into its pseudo-Boolean form and put in list ITEMP. Now it multiplies lists IPRO and ITEMP and puts product in list IMLTMP. After this it returns back.

The flow-charts for these routines are given on the following 3 pages.

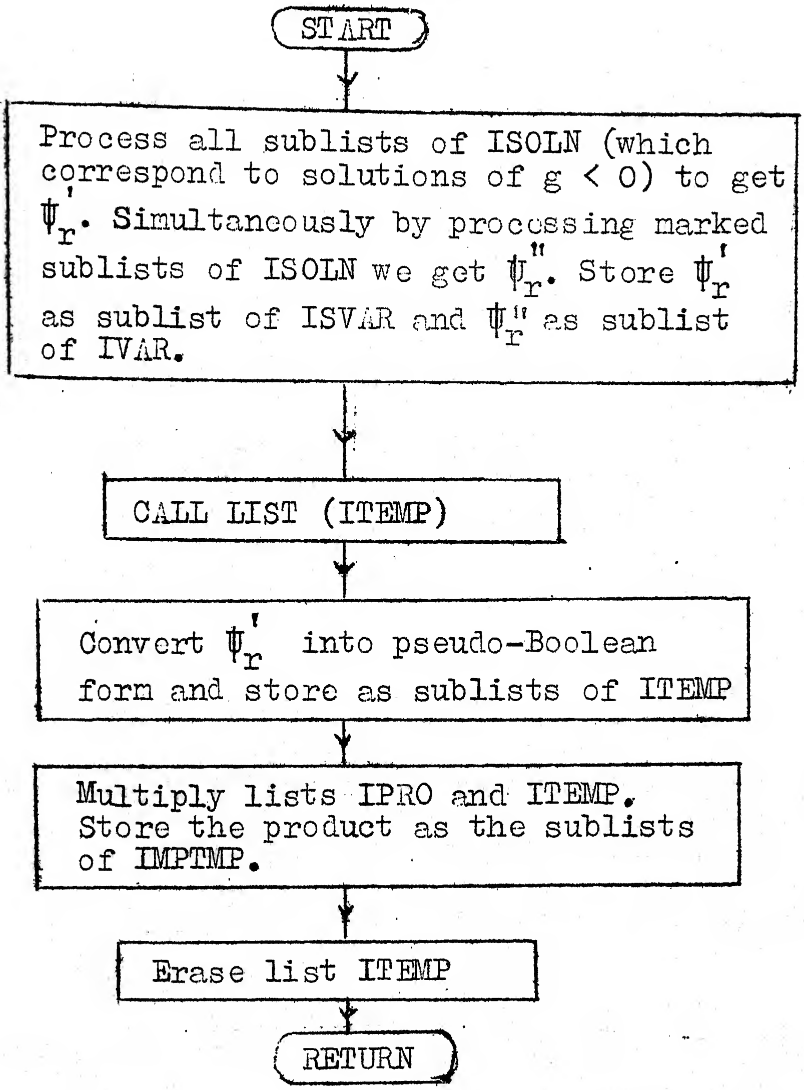
Routine OPTAL



Routine BALGO



Routine ALSOLN



5.2 Comparison of Computational Attributes

Both Maranzana's algorithm and One optimal algorithm do not ensure optimal results. It is quite logical to think, what will happen if we apply these algorithms alternatively as follows:

Let A_r be any solution chosen at random. With A_r as an initial solution let M_r and O_r be the final solutions by Maranzana's algorithm and One optimal algorithm respectivel-

Algorithm:

Step 1: $A \leftarrow A_r$

Step 2: Apply Maranzana's algorithm taking A as initial solution to get B as final solution.

Step 3: If B is same as A, go to Step 6, If not, $A \leftarrow B$ follow Step 4.

Step 4: Apply One optimal algorithm taking A as an initial solution to get B as final solution.

Step 5: If A is same as B, go to Step 6. If not, $A \leftarrow B$, go to Step 2.

Step 6: Stop. B will be the final solution by this procedure.

Among the several computer runs for comparing computational attributes, in most of the cases it was found that B and O_r were same.

In the above algorithm if we interchange the position of Steps 2 and 4, then in all computer runs B was found to be same as O_r.

Table 5.1 summarizes the results of various computer runs for comparing the computational attributes of Maranzana's algorithm and One optimal algorithm. For each entry in columns 3 and 4 mean times are taken over ten different random initial solutions.

If for each entry

M : Mean value of the objective function by Maranzana's algorithm taken over ten random initial solutions $R_1, \ldots, R_{10}$.

O : Mean value of objective function by One optimal algorithm taken over ten random initial solutions $R_1, \ldots, R_{10}$.

M_x : Maximum value of objective function by Maranzana's algorithm taken over ten random initial solutions $R_1, \ldots, R_{10}$.

O_x : Maximum value of objective function by One optimal algorithm taken over 10 random initial solutions $R_1, \ldots, R_{10}$.

Then in column 5

$$\text{Mean error \%} = \frac{M - O}{O} \times 100.$$

In column 6

$$\text{Max error \%} = \frac{M_x - O_x}{O_x} \times 100$$

Conclusions:

(1) The final solution by Maranzana's algorithm was found to be highly sensitive to the initial solution, while by One optimal algorithm in most of the cases the final solution was found to be constant.

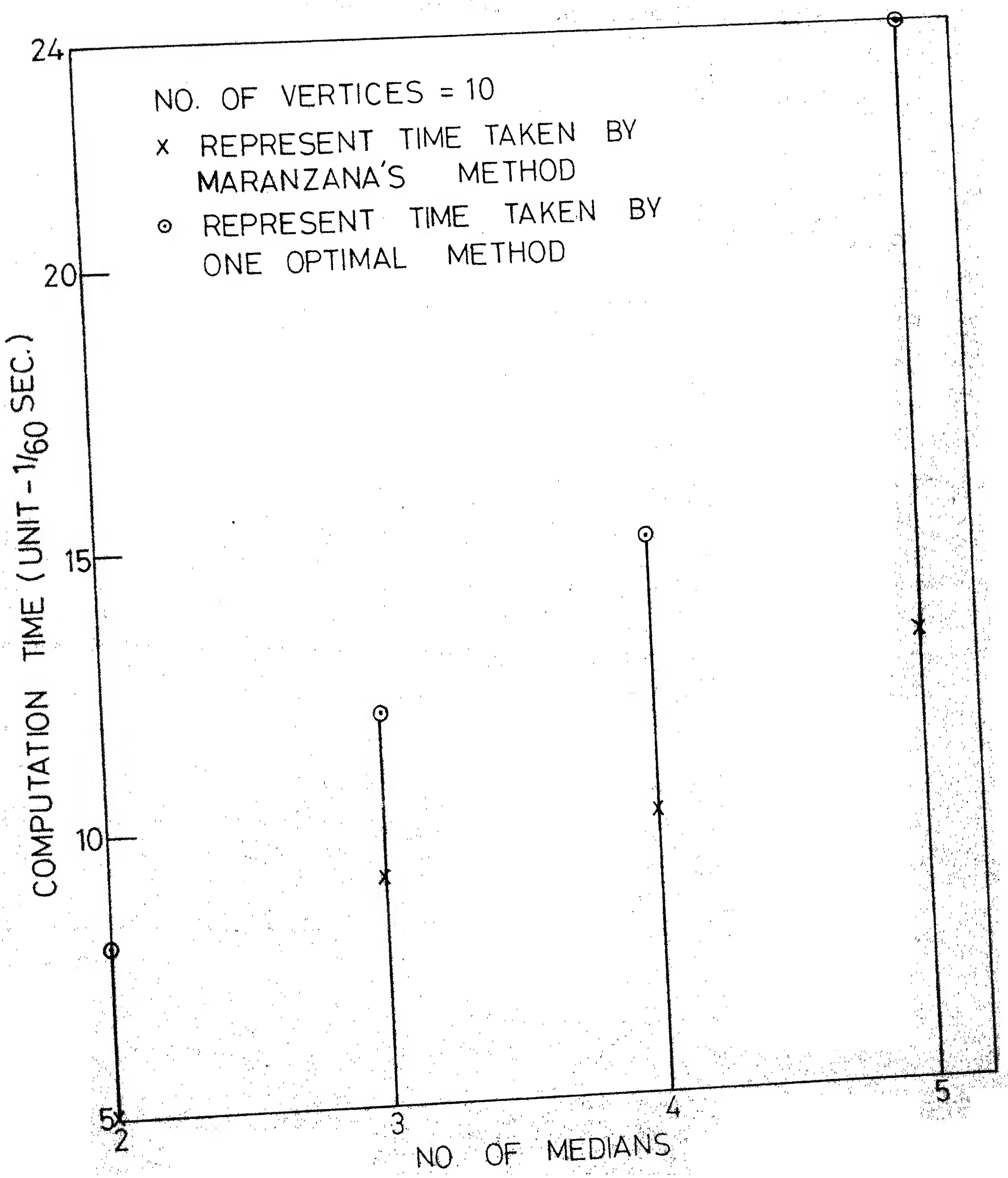
(2) For a given initial solution, the final solution obtained by One optimal algorithm was never found to be inferior to the final solution obtained by Maranzana's algorithm.

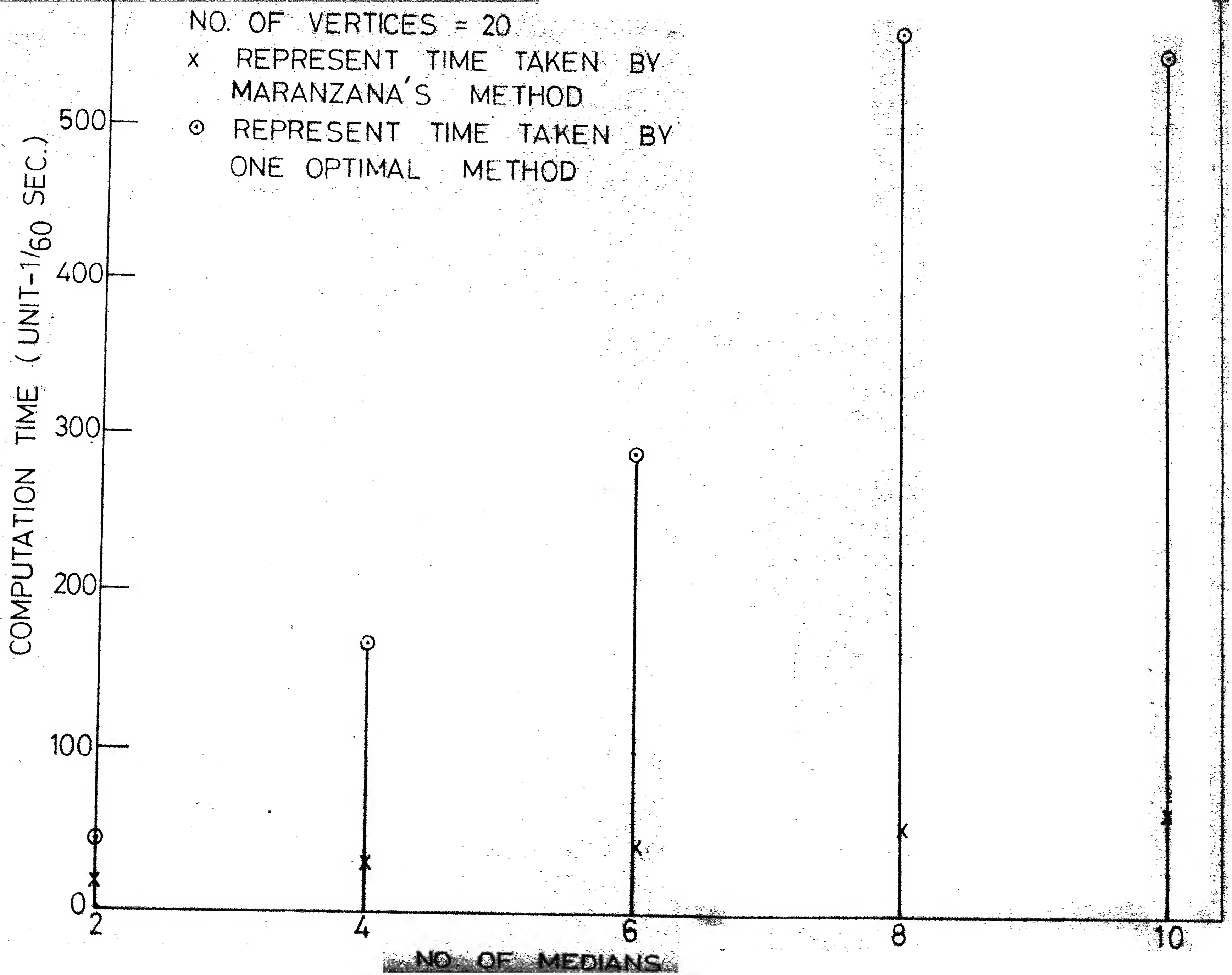
(3) Any One optimal solution taken as initial solution for Maranzana's algorithm was found to give the same final solution.

(4) Since mean error (varying from 6% to 68%) and maximum error (varying from 22% to 206%) were varying between very wide ranges it is very risky to solve the problem by Maranzana's algorithm, although time taken by it may be lesser than that by One optimal algorithm.

No. of vertices n	No. of medians m	Mean computation time for Maranzana's algorithm. Time unit 1/60 sec.	Mean computation time for One-opt. algorithm. Time unit 1/60 sec.	Mean error %.	Max. error %
10	2	5	8	6	37
10	4	10	15	49	112
10	5	13	24	84	194
15	3	13	38	26	105
15	6	24	109	41	88
20	2	13	44	10	24
20	4	27	166	12	22
20	6	40	287	12	29
20	8	52	562	26	74
20	10	61	546	38	57
25	2	22	69	9	26
25	5	60	387	13	49
25	10	107	1024	44	104
30	3	44	292	12	63
30	6	108	1015	33	92
30	9	172	1785	49	75
30	12	186	2538	68	206
30	15	240	2560	67	143

TABLE 5.1


NO. OF VERTICES = 10
x REPRESENT TIME TAKEN BY MARANZANA'S METHOD
⊙ REPRESENT TIME TAKEN BY ONE OPTIMAL METHOD
COMPUTATION TIME (UNIT - 1/60 SEC.)
24
20
15
10
5
2
3
4
5
NO. OF MEDIANS



NO. OF VERTICES = 20
x REPRESENT TIME TAKEN BY MARANZANA'S METHOD
⊙ REPRESENT TIME TAKEN BY ONE OPTIMAL METHOD
COMPUTATION TIME (UNIT-1/60 SEC.)
0
100
200
300
400
500
2
4
6
8
10
NO. OF MEDIANS

REFERENCES

1. Hakimi, S.L.; 'Optimal location of switching centers and the absolute centers and median of a graph', O.R.S.A. 1964, pp. 450-459.

2. Hakimi, S.L.; 'Optimal distribution of switching centres in a communication network and some related graph-theoretic problems', O.R.S.A. 1965, pp.462-475.

3. Hammer, P.L. and S. Rudeanu, 'Pseudo-Boolean programming', O.R.S.A. 1969, pp. 233-261.

4. Hammer, P.L. and S. Rudeanu, 'Boolean methods in operations research', Springer-Verlag 1968.

5. Hammer, P.L. and U.Peled, 'On the maximization of pseudo-Boolean function', JACM, Vol.19, No.2, April 1972, pp. 265-282.

6. Hammer, P.L. and S. Nguyen, 'A partial order in the solution space of bivalent programs', Presented at annual conference of Operations Research Society of America, April 1972.

7. Levy, J.; 'An extended theorem for location on a network', O.R. Quarterly 1967, pp. 433-442.

8. Maranzana, 'On the location of supply points to minimize transportation costs', O.R.Quarterly 1964, pp. 261-270.

9. Teitz, M.B. and P. Bart; 'Heuristic methods for estimating the generalized vertex median of a weighted graph', O.R.S.A. 1968, pp. 955-961.

10. Toreges, C., R. Swain, C. Revella and L. Bergman; 'The location of emergency service facilities', O.R.S.A. September 1971, Vol. 19, pp. 1363-1373.

EE-1972-M-GAR-OPT